МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Липецкий Государственный Технический Университет**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

Лабораторная работа

по программированию №4

“Параметризованные классы”

Студент \_\_\_\_\_\_\_\_\_\_\_\_ Станиславчук С. М.

(подпись, дата)

Группа АС-21-1

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_ Ведищев В. В.

(подпись, дата)

Липецк 2022 г.

Содержание

2. Задание

3. Цель

4. Код программы

5. Вывод

2. Задание

Реализовать на языке C++ параметризированный класс “Матрица”, типы элементов которого могут быть заданы в соответствии с вариантом.

Вариант 11:

Тип 1: вектор в декартовой системе N координат  
Тип 2: вектор в полярных координатах  
Возможность класса: Разность двух матриц и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A-B.

3. Цель работы.

Изучить механизм параметрического полиморфизма на основе создания и использования параметризованных классов.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <cmath>

#include <iomanip>

using namespace std;

double Get\_Rand(double min, double max) {

return (max - min) \* ((double)rand() / (double)RAND\_MAX) + min;

}

template <class T>

class Point {

public:

double x = 0;

double y = 0;

};

class Dekart2D : Point<Dekart2D> {

public:

Dekart2D() {

this->x = 0;

this->y = 0;

}

Dekart2D operator -(Dekart2D other){

Dekart2D res;

res.x = this->x - other.x;

res.y = this->y - other.y;

return res;

}

static Dekart2D rand\_val() {

Dekart2D res;

res.x = Get\_Rand(0, 10);

res.y = Get\_Rand(0, 10);

return res;

}

void to\_str(char\* bufer){

sprintf(bufer, "X=%.2f;Y=%.2f", x, y);

}

};

class Dekart3D : Point<Dekart3D> {

public:

double z;

Dekart3D() {

this->x = 0;

this->y = 0;

this->z = 0;

}

Dekart3D operator -(Dekart3D other) {

Dekart3D res;

res.x = this->x - other.x;

res.y = this->y - other.y;

res.z = this->z - other.z;

return res;

}

static Dekart3D rand\_val() {

Dekart3D res;

res.x = Get\_Rand(0, 10);

res.y = Get\_Rand(0, 10);

res.z = Get\_Rand(0, 10);

return res;

}

void to\_str(char\* bufer) {

sprintf(bufer, "X=%.2f;Y=%.2f;Z=%.2f", x, y, z);

}

};

class Polar : Point<Polar> {

public:

double rho; // magnitude

double theta; // phase angle

double vector;

Polar() {

this->rho = Get\_Rand(0, 9.99);

this->theta = Get\_Rand(-1, 1);

this->vector = rho \* theta;

}

Polar operator -(Polar other) {

Polar res;

res.x = this->x - other.x;

res.y = this->y - other.y;

return res;

}

static Polar rand\_val() {

Polar res;

res.x = res.rho \* acos(res.theta);

res.y = res.rho \* asin(res.theta);

return res;

}

void to\_str(char\* bufer) {

sprintf(bufer, "X=%.2f;Y=%.2f", x, y);

}

};

template <class T>

class Matrix {

T m[4][4];

public:

Matrix() {

for (int i = 0; i < 4; i++)

for (int j = 0; j < 4; j++)

m[i][j] = T::rand\_val();

}

void print() {

char\* bufer = new char[100];

printf("Matrix\n");

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

m[i][j].to\_str(bufer);

printf("%s\t", bufer);

}

printf("\n");

}

printf("\n");

delete[] bufer;

}

Matrix operator - (Matrix& B) {

Matrix<T> res;

for (int i = 0; i < 4; i++)

for (int j = 0; j < 4; j++)

res.m[i][j] = this->m[i][j] - B.m[i][j];

return res;

}

};

template <class T>

void select() {

Matrix<T> m1, m2, m3;

m3 = m1 - m2;

m1.print(); m2.print(); m3.print();

}

int main() {

int choice, CartChoice;

cout << "Type of Matrix is 1:Cartesian 2D, 2:Cartesian 3D or 3:Polar ?: ";

cin >> choice;

switch (choice)

{

case 1: select<Dekart2D>(); break;

case 2: select<Dekart3D>(); break;

case 3: select<Polar>(); break;

default: cout << "Unexpected choice\n"; break;

}

return 0;

}

Вывод: изучил механизм параметрического полиморфизма на основе создания и использования параметризованных классов.